UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF NEW YORK

----------X

LUCIANO F. PAONE,

Plaintiff,

-against-

MICROSOFT CORPORATION,

Defendant.

----------X

MEMORANDUM OF DECISION AND ORDER
07-cv-2973 (ADS)(ARL)

APPEARANCES:

Kirkland & Ellis LLP
Attorneys for the plaintiff
153 East 53rd Street
New York, NY 10022
By: Andrew Gordon Heinz, Esq.,
Jeanne M. Heffernan, Esq.,
John Michael Desmarais, Esq.,
Jon Todd Hohenthaner, Esq.,
Ryan Charles Micallef, Esq., of Counsel

Woodcock Washburn LLP
Attorneys for the defendant
2929 Arch Street
12th Floor
Philadelphia, PA 19104-2891
By: Dale M. Heist, Esq.,
Daniel J. Goettle, Esq.,
John E. McGlynn, Esq.,
Steven J. Rocci, Esq., of Counsel

Westerman, Ball, Ederer, Miller & Sharfstein, LLP
Attorneys for the defendant
1201 RXR Plaza
Uniondale, NY 11556
By: Greg S. Zucker, Esq.,
Jeffrey A. Miller, Esq., of Counsel

Special Master Gale R. Peterson, Esq.
Cox & Smith Incorporated
112 E. Pecan Street
Suite 1800
San Antonio, TX 78205-1521

SPATT, District Judge.

In this patent infringement case, the plaintiff Luciano F. Paone alleges that the defendant Microsoft Corporation ("Microsoft") has infringed United States Patent 6,259,789 ("the '789 Patent"), which Paone holds. Pursuant to the Supreme Court's decision in Markman v. Westview Instruments, Inc., 517 U.S. 370, 116 S. Ct. 1384, 134 L. Ed. 2d 577 (1996), the Court now construes the disputed claim terms of the '789 Patent.

I. BACKGROUND

A. Background of the Invention

The United States Patent and Trademark Office ("PTO") issued the '789 Patent, entitled "Computer Implemented Secret Object Key Block Cipher Encryption and Digital Signature Device and Method", to the plaintiff Luciano F. Paone on July 10, 2001. The '789 Patent describes a method of translating (or "encrypting") ordinary data (called "plaintext") into encoded data (called "ciphertext"), so that the plaintext may not be viewed by an unintended reader. In cryptography, which is the science of encryption, such a method is called a "cipher". Generally, encrypted ciphertext is later decoded, or "decrypted", into plaintext, so that the data is again usable. The cipher described in the '789 Patent is a computer based "symmetric key cipher."

By way of background, a very simple cipher (that is, method) used to encrypt a message sent in English might be to replace each letter in the plaintext with the character that directly follows that letter in the alphabet. Using that method, each "a" in the message would be replaced with a "b", each "b" in the message would be replaced with a "c", each "c" would be replaced with a "d", and so forth. Each "z" would be replaced with an "a". Using this cipher, the message "Hello there" becomes "Ifmmp uifsf." A person who knows the method of encryption—that is, who knows the cipher—can easily decode the message and read it.

A cipher that uses a "symmetric key" requires the person encrypting and the person decrypting to know an additional piece of information to use the cipher. For example, we might have a cipher that calls for the replacement of each letter of plaintext with a character that follows that letter *some number* of letters later in the alphabet. This type of cipher establishes a pattern, but it doesn't provide all of the information needed to encrypt or decrypt data. Here, the pattern is that each letter of the plaintext will be replaced a letter that appears a given number of characters later on in the alphabet. The key is *how many letters later* in the alphabet will be found each letter's replacement. The following is an example of the operation of this cipher:

[Remainder of page intentionally left blank.]

Plaintext	**Cipher (method)**	**Key (in this cipher, how many characters later in the alphabet is the replacement character?)**	**Ciphertext**
Hello there	Replace each plaintext letter with a character that appears some number of characters later in the alphabet.	2 characters later	Jgnnq vjgtg
Hello there	(same)	3 characters later	Khoor wkhuh
Hello there	(same)	4 characters later	Lipps xlivi

Under this system, a person who knows just the cipher (that is, who knows that letters in the message will be replaced with characters appearing *some* number of letters later in the alphabet) cannot decrypt the ciphertext, because that person doesn't know the key. But a person who knows both the cipher and the key can easily decode the message.

Of course, this example is a very simple symmetric key cypher, and a person without knowledge of either the cipher or the key could probably guess an encoded message if given time. Thus, to defend against this, cryptologists use far more complicated methods of encryption. Moreover, with the advent of the computer, the ability—and the need—to encrypt with great sophistication has increased dramatically. In keeping with this trend, the '789 Patent focuses on computer-based encryption of data.

Generally, when computers encrypt data, they do not directly encrypt English language text, as in the example above. Rather, text is first translated into a long stream of 1's and 0's—each 1 or 0 being called a "bit", and eight bits being called a

"byte"—which can later be changed back into text. Other types of computer data, such as images or spreadsheets, are also translated into 1's and 0's before encryption. Generally, the binary code (that is, the 1's and 0's) that represents readable data is insecure before encryption, because any computer with the correct software could translate these bits back into the original text, image, or spreadsheet. Therefore, if a person wants to send or store sensitive data without fear of it being compromised, the stream of 1's and 0's must be altered to prevent the files from being viewed by unintended readers.

There are several ways to encrypt computer data, but the '789 Patent deals exclusively with "symmetric key" encryption of computer data, which as noted above, requires that the encryptor and decryptor share knowledge of both a cipher and a key. Computer based symmetric key ciphers divide roughly into two types: block ciphers and stream ciphers. As the distinction between these two types of ciphers forms a major ground for dispute between the parties, the Court discusses this issue in much greater detail below. However, as a general matter, a stream cipher employs its key to encrypt data one bit at a time, while a block cipher employs its key to encrypt bits in groups. The '789 patent describes a block cipher.

Computer implemented block ciphers have been in wide use in the United States since at least 1976, when a block cipher called the "Data Encryption Standard" or "DES" was adopted by the United States government for general use. As of 1997, when Paone filed his patent application, dozens of additional block encryption algorithms had been published. However, most of those inventions used a single key

to encrypt successive blocks of data. Paone's innovation—in the most general terms—was to change the encryption key for each data block, based on additional, randomly generated data.

B. Procedural History

Paone filed his initial application for the present patent on December 12, 1997. Following that initial application, the PTO rejected Paone's claims as unpatentable in three successive office actions, dated September 29, 1999, March 16, 2000, and October 30, 2000. In response to each of these rejections, Paone modified his claims and provided additional argument. Then, on July 10, 2001, the PTO deemed the described invention patentable, and issued the '789 Patent.

On July 23, 2007, Paone commenced the present action against Microsoft, alleging that Microsoft was infringing the '789 patent. Specifically, Paone asserts that two components of Microsoft's flagship computer operating system, Windows, infringe the '789 patent. The parties proceeded with discovery, during which time Microsoft on May 16, 2008 requested the PTO to reexamine the '789 patent. In early 2009, with the reexamination proceeding still pending, the parties briefed claim construction motions. However, before holding a Markman hearing, the Court on April 5, 2009 stayed the case pending the resolution of the reexamination proceeding. Then, before the stay was lifted, Microsoft also filed two more reexamination requests, dated June 29, 2009, and July 27, 2009. The three reexamination proceedings as a whole resulted in the cancellation of claims 1, 3, 23, and 32 of the '789 patent, but also a ruling that several claims, including claims 2, 24, 33, and 34,

were patentable. A number of other claims were also found to be patentable after certain amendments.

After the reexamination proceedings had been finalized, the Court on March 3, 2010 lifted the stay of the case. Partly as a result of the reexamination proceedings, Paone modified his position to assert that Microsoft was infringing claims 2, 24, 33, and 34 of the '789 patent, all of which had been ruled patentable by the PTO. These claims read in full as follows (the language of claims 1, 23, and 32 are also included, because some of the asserted claims are dependent on those claims).

What is claimed is:

> **1.** A computer implemented method for encrypting data comprising the steps of:
>
> creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data;
>
> creating a key schedule based upon the at least one object key;
>
> encrypting a random session object key in a block cipher encryption process with the at least one object key;
>
> encrypting a block of input plaintext data utilizing said key schedule;
>
> modifying the at least one object key based on seeding from the random session object key;
>
> modifying the key schedule based upon the at least one modified object key;
>
> encrypting a next block of input plaintext data utilizing said modified key schedule; and
>
> repeating the steps of modifying the at least one object key, modifying the key schedule and encrypting utilizing the modified

key schedule until the encrypting of blocks of plaintext data is completed.

2. A computer implemented method as defined in claim 1, wherein the modification of the key schedule is independent of the input plaintext data.

23. A cryptographic communications system comprising:

at least two networked computer systems linked by a communication channel; and

each computer system including a central processing unit and a memory storage decive for executing a block cipher encryption/decryption process;

wherein the encryption process transforms an input plaintext message to a ciphertext message and the decryption process transforms the ciphertext message to the input plaintext message, the encryption/decryption process using at least one dynamic object key which is modified using a non-linear function for each block of input data, each object key being associated with a different key schedule to encrypt/decrypt the input plaintext/output ciphertext message.

24. A cryptographic communications system as defined in claim 23, wherein the encryption/decryption process further includes the use of a random session object key having an initial state randomly generated by the computer system, and wherein the object key modifications are based on seeding from the random session object key.

32. A computer implemented method for encrypting data comprising the steps of:

creating at least one object key in a block cipher, the at least one object key comprisingdata and methods that operate on said data;

creating a key schedule based upon the at least one object key;

encrypting a block of input plaintext data utilizing said key schedule;

modifying the at least one object key based on at least a non-linear function;

> modifying the key schedule based upon the at least one modified object key;
>
> encrypting a next block of input plaintext data utilizing said modified key schedule; and
>
> repeating the steps of modifying the at least one object key, modifying the key schedule and encrypting utilizing the modified key schedule until the encrypting of blocks of plaintext data is completed.

33. A computer implemented method a defined in claim 32, where in the non-linear function is a hashing function.

34. A cryptographic communications systems [*sic*] as defined in claim 23, wherein the non-linear function is a hashing function.

Upon the lifting of the stay, the parties resumed their claim construction motions. Although the parties briefed claim construction before the case was stayed, the parties also filed supplemental claim construction briefs to address the reexamination proceedings. Then, on the consent of the parties, the Court appointed Gale R. Peterson, Esq., an experienced and well-regarded patent attorney, as a special master in this case for the limited purpose of conducting a Markman hearing and issuing a report and recommendation to the Court on claim construction. On June 23, 2010 Special Master Peterson conducted a non-evidentiary Markman hearing, at which the parties' attorneys presented their arguments on claim construction. On August 11, 2010, Special Master Peterson issued an extensive and detailed Report and Recommendation on claim construction (the "Special Master's Report"). In response to this report, both Paone and Microsoft filed memoranda of law on September 17, 2010, each objecting in part to the Special Master's Report.

Prior to the Markman hearing, the parties presented nine claim terms for construction. However, after meeting and conferring at the suggestion of the Special Master, the parties agreed on meanings for three of those terms. The remaining six terms in dispute are:

Disputed Terms/Phrases	**Asserted Claims (or underlying independent claims) Containing the Disputed Terms/Phrases**
"object key"	1, 23, 24, 32
"random session object key"	1, 24
"repeating the step[] of modifying the at least one object key"	1, 32
"block"	1, 23, 32
"key schedule"	1, 2, 23, 32
"block cipher"	1, 23, 26, 32

In light of the parties' submissions and the Special Master's Report, the Court now rules on the construction of these six claim terms.

II. DISCUSSION

A. Standard on Review of a Special Master's Report and Recommendation

Fed. R. Civ. P. 53(f)(1) provides that, "[i]n acting on a master's order, report, or recommendations, the court . . . may adopt or affirm, modify, wholly or partly reject or reverse, or resubmit to the master with instructions." Rule 53(f)(4) further states that "[t]he court must decide de novo all objections to conclusions of law made or recommended by a master." Under Markman, 517 U.S. at 290, claim construction is a matter of law. Therefore, the Court reviews the Special Master's recommendations *de novo*.

B. Principles of Claim Construction

The Special Master provided a helpful and detailed recitation of the core principles of claim construction in his Report, and the Court adopts that discussion in full. (See Special Master's Report at 5–10.) In discussing these principles, the Special Master quoted extensively from the Federal Circuit's leading case on the law of claim construction, Phillips v. AWH Corp., 415 F.3d 1303 (Fed Cir. 2005). Significant sections of that case bear repeating here. In Phillips, the Federal Circuit stated:

> It is a bedrock principle of patent law that the claims of a patent define the invention to which the patentee is entitled the right to exclude. . . . We have frequently stated that the words of a claim are generally given their ordinary and customary meaning. We have made clear, moreover, that the ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, i.e., as of the effective filing date of the patent application. . . . The inquiry into how a person of ordinary skill in the art understands a claim term provides an objective baseline from which to begin claim interpretation. . . . Importantly, the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification. . . . The inventor's words that are used to describe the invention—the inventor's lexicography—must be understood and interpreted by the court as they would be understood and interpreted by a person in that field of technology. . . .
>
> In some cases, the ordinary meaning of claim language as understood by a person of skill in the art may be readily apparent even to lay judges, and claim construction in such cases involves little more than the application of the widely accepted meaning of commonly understood words. . . . In many cases that give rise to litigation, however, determining the ordinary and customary meaning of the claim requires examination of terms that have a particular meaning in a field of art. Because the meaning of a claim term as understood by persons of skill in the art is often not

immediately apparent, and because patentees frequently use terms idiosyncratically, the court looks to those sources available to the public that show what a person of skill in the art would have understood disputed claim language to mean. Those sources include the words of the claims themselves, the remainder of the specification, the prosecution history, and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art. . . .

Other claims of the patent in question, both asserted and unasserted, can also be valuable sources of enlightenment as to the meaning of a claim term. Because claim terms are normally used consistently throughout the patent, the usage of a term in one claim can often illuminate the meaning of the same term in other claims. . . .

The claims, of course, do not stand alone. Rather, they are part of a fully integrated written instrument, consisting principally of a specification that concludes with the claims. For that reason, claims must be read in view of the specification, of which they are a part. As we stated in Vitronics, the specification is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term. . . . In light of the statutory directive that the inventor provide a full and exact description of the claimed invention, the specification necessarily informs the proper construction of the claims. Consistent with that general principle, our cases recognize that the specification may reveal a special definition given to a claim term by the patentee that differs from the meaning it would otherwise possess. In such cases, the inventor's lexicography governs.

In addition to consulting the specification, we have held that a court should also consider the patent's prosecution history, if it is in evidence. The prosecution history, which we have designated as part of the intrinsic evidence, consists of the complete record of the proceedings before the PTO and includes the prior art cited during the examination of the patent. . . . Yet because the prosecution history represents an ongoing negotiation between the PTO and the applicant, rather than the final product of that negotiation, it often lacks the clarity of the specification and thus is less useful for claim construction purposes. Nonetheless, the prosecution history can often inform the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be. . . .

> Although we have emphasized the importance of intrinsic evidence in claim construction, we have also authorized district courts to rely on extrinsic evidence, which consists of all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises. However, while extrinsic evidence can shed useful light on the relevant art, we have explained that it is less significant than the intrinsic record in determining the legally operative meaning of claim language. . . .
>
> Within the class of extrinsic evidence, the court has observed that dictionaries and treatises can be useful in claim construction. . . . Because dictionaries, and especially technical dictionaries, endeavor to collect the accepted meanings of terms used in various fields of science and technology, those resources have been properly recognized as among the many tools that can assist the court in determining the meaning of particular terminology to those of skill in the art of the invention. . . .
>
> We have also held that extrinsic evidence in the form of expert testimony can be useful to a court for a variety of purposes, such as to provide background on the technology at issue, to explain how an invention works, to ensure that the court's understanding of the technical aspects of the patent is consistent with that of a person of skill in the art, or to establish that a particular term in the patent or the prior art has a particular meaning in the pertinent field. However, conclusory, unsupported assertions by experts as to the definition of a claim term are not useful to a court.
>
> We have viewed extrinsic evidence in general as less reliable than the patent and its prosecution history in determining how to read claim terms, for several reasons. First, extrinsic evidence by definition is not part of the patent and does not have the specification's virtue of being created at the time of patent prosecution for the purpose of explaining the patent's scope and meaning. Second, while claims are construed as they would be understood by a hypothetical person of skill in the art, extrinsic publications may not be written by or for skilled artisans and therefore may not reflect the understanding of a skilled artisan in the field of the patent. . . . Finally, undue reliance on extrinsic evidence poses the risk that it will be used to change the meaning of claims in derogation of the indisputable public records consisting of the claims, the specification and the prosecution history, thereby undermining the public notice function of patents.

415 F.3d at 1312–1319 (internal citations and quotations omitted, paragraphing is altered from original). With these basic principles in hand, the Court proceeds to construe the disputed claim language.

C. Construction of the Claim Term "Object Key"

The first term that the Court construes is "object key", which appears throughout the patent's claims and specification, as well as in the patent's title. The parties agree that the term "object key" refers to a key that is used to encrypt data, and they also agree that this term did not have a standard meaning within the fields of computer science or cryptography prior to the filing of the '789 patent. The use of "object key" throughout the patent is typified by its use in claim 1, which states that one of the methods of the invention is "creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data." ('789 Patent, 11:19–21; see also, e.g., id. at 18:12–14.) The parties and the Special Master construed "object key" as follows:

[Remainder of page intentionally left blank.]

Paone	Microsoft
"data and methods that operate on the data" [Does not object to the Special Master's recommendation.]	"self-contained module, defined in an object-oriented program, having a secret code for controlling encryption" [Objects to the Special Master's recommendation.]
Special Master Peterson	
In the field of cryptography, encryption and decryption are controlled by keys. The inventor coined the term "object key" to refer to a "first" encryption key which is distinct from a "second" encryption key which the inventor called a "random session object key," also a coined term. Both are "dynamic," i.e., both are modified from an initial static state, however it is unnecessary to expressly add that to the construction of "object key" because that modification is required by other claim language. The term "object key" per se as used in the claims and specification, simply means a "first encryption key."	

Paone and Microsoft dispute two issues with respect to the construction of "object key." The first, less contentious issue is whether the "object key" must be a "secret". The second, more contentious issue is whether the "object key" must be defined in "object-oriented programming" and be "self-contained." As the Special Master's construction indicates, he found in favor of Paone on both of these issues, ruling that the "object key" need neither be "secret" nor defined in object-oriented programming and self-contained.

1. Must the Object Key be "Secret"?

Neither the parties, the intrinsic evidence, nor the Special Master address in great detail whether the "object key" must be "secret." Discussing this issue in his Report, the Special Master stated:

> Regarding [whether the object key must be] "secret," although the specification does refer to a "secret" object key, e.g., "a secret key composed of two 2048-bit user object keys," and "a 4096-bit secret key block cipher data encryption process," the claims do not require

> that the "object key" be "secret." During the Markman hearing, counsel for Paone explained that "secret" was not a necessary characteristic of an "object key." Tr. at 12-14. That is, whether an "object key" was "secret" or not, would affect the security of the system, but was not a necessary attribute of an "object key" per se. Also, at least the first two "objects of the invention" listed in the '789 patent refer to "object key" with no requirement that the "object key" be "secret." Although the issue is close, the master believes that in this instance, the plain language of the claim controls. That is, requiring that the "object key" be "secret" crosses the line from interpreting claims "in light of" of the specification, to incorporating terms from the specification into the claims, which, of course, is improper. The "object key" may be, and perhaps preferably is, "secret," but the claims do not expressly require the same. The specification also does not say that the "object key" must be "secret," at least expressly. Moreover, it was unclear from the Markman hearing whether, and to what extent, "secret" has any impact on the ultimate question of infringement. On the current record, therefore, the master concludes that the claims do not require that an "object key" be "secret." Or, in other words, a "key" otherwise meeting the terms of the claims is not excluded because it is not "secret." However, that is subject to potential reconsideration.

(Special Master's Report at 44, fn. 15.)

The Special Master's points on the issue are well taken. Specifically, it is important that, while the title and specification for the patent refer to a "secret" object key, the claims themselves neither implicitly nor explicitly state that the "object key" must be secret. Nevertheless, reading the claims "in view of the specification," Markman v. Westview Instruments, Inc., 52 F.3d 967, 978 (Fed. Cir. 1995), the Court concludes that the "object key" must be secret. Not only is the object key referred to as "secret" in several places in the specification and the patent title, but the "Background of the Invention" portion of the specification focuses almost exclusively on the importance in cryptography of maintaining the secrecy of an encrypted message. Specifically, Paone describes the invention in that section stating:

> Accordingly, the present invention seeks to overcome the disadvantages associated with currently available encryption methods and create ciphertext which is very secure and substantially immune to known cryptanalytic attacks.

('789 patent, 2:13–17.) Paone's attorney conceded at oral argument that a non-secret object key would affect the security of an encryption system, and it appears to the Court that a non-secret object key would undermine the usefulness of the entire invention. As such, reading the claims in light of the specification, the Court concludes that the "object key" must be "secret," in the sense that it cannot be available to the general public. However, in deference to the claim language itself, the Court declines to provide a more limiting definition of "secret."

2. Must the Object Key be Defined in Object-Oriented Programming?

The more central argument over the meaning of "object key" is whether the object key must be defined in what computer scientists call "object-oriented programming". Concomitant with this dispute is whether the object key must be also "self-contained". This disagreement is so contentious that the Special Master's resolution of it consumes approximately 109 pages of the his 243-page Report. Thus, to fully understand the parties' disagreement, a brief background discussion of object-oriented programming is in order.

Object-oriented programming ("OOP") is not easy to define, and a recently published book for beginning computer programmers notes that "OOP is difficult to summarize because it doesn't represent a single concept . . . and even experts are unable to agree on a common definition." Richard Mansfield, Programming: A Beginner's Guide, p. 264 (McGraw Hill 2009). Even the programming language

C++, one of the most widely used object-oriented programming languages, is viewed by some as "not a pure OOP language." Namir C. Shammas, Foundations of C++ and Object-oriented Programming, p. 10 (IDG Books Worldwide, Inc. 1995).

However, on a very basic level, object-oriented programming is usually defined in contrast to a method of computer programming that preceded OOP, which is called "structured" (or "functional") programming. Structured programming is a method of giving computers instructions in "reusable subroutines and functions . . . [which] enjoy[] a certain level of autonomy". Shammas at 7. Thus, structured programs can define a complicated task for a computer to do—say, calculate the square root of a number—and define that task in a "subroutine". That subroutine contains all of the steps necessary to do the task, so the task can repeated over and over by using the subroutine, without forcing the programmer to re-write the individual steps each time.

However, structured programming generally stores data in ways that permit data to be manipulated broadly, which "[i]n large-scale software projects . . . can lead to chaos." See Shammas at 7, 9. In other words, this means that in structured programming, the data used by a given subroutine is shared with and can be modified by other subroutines. One result of this is that when data becomes damaged or lost, it can be difficult to determine the source of the problem because so many different subroutines have access to the data.

Object-oriented programming seeks to solve this problem. While an object-oriented program still defines complex tasks into subroutines, it allows only the piece

of the program that uses each piece of data most directly to have immediate access to that data. Thus, in object-oriented programming, instead of using subroutines as the building blocks of a program, programmers use "objects". "Objects" consist of *both* the instructions for various subroutines *and* the data needed to carry out those instructions. Thus, whereas the data in a structured program might be very loosely thought of as being in a shared pool, the data in an object-oriented program is encapsulated into pockets, accessible only by the "object" that will use that data directly. Any other part of the program that wishes to change an object's data must go through the object to do so. Objects are thus described variously as having "attributes and [] methods that alter these attributes," Shammas at 15; "both data . . . as well as code that processes that data," Mansfield at 264; or "data and the operations that manipulate the data," Matt Weisfeld, The Object-Oriented Thought Process, p. 9 (Sams Publishing 2004). (The Court notes that while some of these sources post-date the filing of the '789 patent, there is no indication that the understanding of this basic theory in object-oriented programming has changed since the '789 patent was filed in 1997.) Generally, a programming language is viewed as either a "structured programming language" or an "object-oriented programming language"—although there are some exceptions to this rule.

Returning, then, to the construction of the term at hand, Microsoft urges that the "object key" must be defined in an object-oriented program, while Paone asserts that "object key" is explicitly defined in the claims as "data and methods that operate on said data". Microsoft bases its conclusion primarily on (1) Paone's use of the

word "object" in naming the claim term, and (2) Paone's repeated indication that the object key "comprises data and methods that operate on said data." (See, e.g., '789 Patent, 11:20–21.) Microsoft asserts that, in light of the state of the art in computer science, using the term "object" with a consistent indication that the element described contains both data and methods would indicate to a person skilled in computer science that the "object key" is intended to be executed in an object-oriented program. Microsoft contends that this conclusion is confirmed by the fact that more than one patent examiner who examined the '789 Patnet understood that the term "'object key comprising data and methods' . . . refer[s] to an object associated with object-oriented programming." (See Paone's Claim Construction Brief at Ex. B, PAONE0003100.) Microsoft also points out that Paone never directly challenged this conclusion.

Once Microsoft concludes that an "object key" must be defined in an object-oriented program, Microsoft then points to three technical dictionaries that define an "object" in object-oriented programming as "self-contained". Based on this, Microsoft also maintains that an object key must be "self-contained".

After examining both the intrinsic and extrinsic evidence presented, the Special Master rejected Microsoft's construction of "object key" as requiring that this element be executed in object-oriented programming or be self-contained. While the Special Master also rejected Paone's assertion that "object key" was explicitly defined in the claims, he essentially ruled in Paone's favor by not limiting the term as Microsoft proposed.

The Special Master based his ruling first and foremost on the fact that *nowhere* in the patent specification, the claims, or in the prosecution history does Paone explicitly assert that his invention is limited to execution in object-oriented programming. While Microsoft argues that Paone made a significant admission when he failed to contest the examiners' assumption that the object key would be defined in object-oriented programming, Microsoft otherwise does not contest that the term "object-oriented programming" appears nowhere in the patent or in Paone's submissions to the PTO. Likewise, the Special Master noted that there was no indication that the use of object-oriented programming rendered patentable the subject matter of the '789 patent. (See Special Master's Report at 95 ("If the inventor's belief had truly been – as Microsoft suggests – that his contribution and distinguishing feature over the prior art was the use of object-oriented programming, it would stand to reason (or, at least seemingly so) that the inventor would have included some mention of object-oriented programming in at least the specification, if not expressly in the claims."); see also id. at 123 ("it is clear that the patentability of the asserted claims was not based on limiting 'object key' to implementation using object-oriented programming").)

As for the prosecution history of the '789 patent, the Special Master provided a detailed overview of the prosecution and reexamination proceedings, and determined that the examiners' conclusion that "object key" implied the use of object-oriented programming did not require that the patent be so limited. First, although Paone never explicitly challenged this conclusion, he also never disclaimed that his

patent would cover object keys executed in non-object-oriented programming. Moreover, the Special Master noted that the examiner who first suggested that the use of object-oriented programming was implicated did so by interpreting the word "object" in the context not of computer science generally, but in the context of object-oriented programming. The Special Master then noted that "[e]ven in the field of computer science, insofar as understood, 'object' only begins to take on a special connotation if the field is narrowed to object-oriented programming." (Id. at 106.) Finally, the Special Master concluded that "here one reading the entire history of prosecution would see that the applicant expressly informed the PTO that 'object key' was not limited to implementation using object-oriented programming." (Id. at 124.)

Based on this, the Special Master concluded that the word "object" in "object key" connoted no special meaning, and that the "term is no different from 'first key' or 'widget key' or any other coined term." (Id. at 34.) The Special Master rejected Microsoft's proposed dictionary definitions of "object" as being definitions "in object-oriented programming," and not definitions of "object" in computer science generally. (See id. at 39 ("each definition [of 'object' presented by Microsoft] begins, '[i]n object-oriented programming.' Again, those definitions establish little more than 'object' has a recognized meaning in the field of object-oriented programming, and do not establish that one of ordinary skill in the art seeing 'object key' in the claims and specification of the '789 patent would, on that basis alone, construe "object key" as limited to object-oriented programming.").) These dictionaries provided the source

for limiting the object key to being "self-contained", and with the rejection the dictionaries' definitions necessarily followed the rejection of this limitation.

Ultimately, the Special Master recommended that the Court find that the "object key" is merely the "first encryption key," to be distinguished from the "random session object key," which is the "second encryption key." The Special Master did not address what it would mean that the "object key" should comprise both (1) data and (2) methods that operate on that data, but he did note that other claim terms provided that the object key was necessarily "dynamic," and that it was "modified from an initial static state." (Id. at 124.)

In reviewing the Special Master's Report, the Court agrees with the Special Master that an "object key" need not be executed in an object-oriented program and need not be self-contained. However, for the reasons that follow, the Court respectfully disagrees with the Special Master's determination that the word "object" in the claim term connotes no special meaning in the context of the patent.

First, with respect to the Court's partial adoption of the Special Master's recommendation, the Court agrees that Paone makes no explicit reference to object-oriented programming in any part of the patent. In addition, the Court agrees with and adopts the Special Master's rulings on the import of the prosecution history with respect to the construction of the term "object key", and agrees with the Special Master that the parties' experts' reports are both entitled to little weight. Thus, granting Paone the "full scope of [his] claim language," Home Diagnostics, Inc. v. LifeScan, Inc., 381 F.3d 1352, 1358 (Fed Cir. 2004), the Court finds that Paone's

implicit reference to object-oriented programming does not limit the technical execution of the invention to an object-oriented programming language.

As for the Court's partial rejection of the Special Master's recommendation, the Court notes that one of the most basic concepts of object-oriented programming is to create programs based on "objects" that contain both data and methods that operate on that data. Having reviewed both the intrinsic and extrinsic evidence, the Court finds that a person skilled in the art of computer science—who would necessarily have been aware in 1997 that object-oriented programming was the dominant paradigm of commercial computer programming—would recognize Paone's construction as a reference to object-oriented programming. As such, Paone's choice of the word "object" to modify "key" carries greater meaning than a purely coined term, such as "widget key" or "first key".

While the Court rejects the limitation that the object key must be executed in an object-oriented program, the Court holds that "object key" refers to an element of the invention that functions analogously to an object in an object-oriented program. Just like a *bona fide* object in an object-oriented program, the "object key" contains both data—the key data that will inform the operation of the cipher—and the methods that operate on that data. More importantly, just as in an object-oriented program, the methods contained in the object key are the *only* methods that operate on the key data.

Here, while the nature of those methods comprised in the object key are not explicitly stated, the Court can readily deduce them. The Court begins with the general principal of claim construction that "claims are interpreted with an eye toward

giving effect to all terms in the claim." Bicon, Inc. v. Straumann Co., 441 F.3d 945, 950 (Fed. Cir. 2006); accord Cohesive Technologies, Inc. v. Waters Corp., 543 F.3d 1351, 1368 (Fed. Cir. 2008). In the '789 Patent, the Court must give meaning to, among other things, language in claims 1 and 32 that states that the object key comprises "data and methods that operate on said data". In the Court's view, this language would be superfluous if those methods were never utilized by the described invention. Thus, looking to the other language describing the invention, Claims 1 and 32 disclose only one operation that is performed on the key data: modification of that data for each plaintext block. Thus, to give meaning to "methods that operate on said data", the Court construes this language to refer to methods that modify the object key's data. This construction is consistent with other claims in the patent, as well as the patent specification. (See '789 Patent, 13:34–35 (describing in claim 14 "the modification method *of* the object key" (emphasis added)); 5:18–20 ("[t]he object key [] contains key modification methods that mutate the key's state."); 6:45–47 ("the object keys [] are composed of initial states that are created by the user and functions that modify the keys for each plaintext input block"); 7:4–9 ("After each block has been processed, the key modification methods in the object key . . . are performed to modify the object key's . . . state").)

Nevertheless, as a matter of technology, there is no indication that this conceptual construction must be carried out in an object-oriented programming language. The claim language and specification say nothing that indicates that the invention must be practiced using a computer programming language that *prevents*

other parts of the program from modifying the object key's data, such as an object-oriented programming language would do. Indeed, the patent does not even suggest that this would benefit the invention.

The Court's construction is consistent with the Federal Circuit's interpretation of the term "object" in a different patent involving computer science in TiVo, Inc. v. EchoStar Communications Corp., 516 F.3d 1290, 1306–07 (Fed. Cir. 2008). While a single word need not have the same meaning in two different patents, the Federal Circuit's reasoning in TiVo concerning the word "object" comports with this Court's reasoning on that issue here. In TiVo, the Federal Circuit reviewed the construction of several claim terms involving computer software that each included the word "object". The court held that the use of the word "object" did not require the use of an object-oriented programming language in practicing the invention, and stating:

> While the patent specification includes an embodiment showing the use of "a C// class hierarchy derivation of the program logic," '389 patent, col. 8, ll. 9-10, and uses terms characteristic of object-oriented programming in connection with that example, neither the written description nor the claims anywhere state or imply that the invention must use object-oriented programming in general, or C// in particular. Without more, the use of an example that employs object-oriented programming is not sufficient to require that the claims be limited to embodiments using C// or a similar programming language.

Id. at 1307. Here, unlike TiVo, Paone has not even presented an example of the '789 subject matter that is practiced using an object-oriented programming language. Thus, like the TiVo court, this Court finds that the use of the term "object" in the context of language typical of object-oriented programming does not require that the invention be practiced in an object-oriented programming language.

As for the Microsoft's request that "object key" be defined as "self-contained," the Court agrees with the Special Master that the technical dictionaries from which Microsoft derives this definition beg the question. The Special Master points out that each definition of "object" that Microsoft relies on explicitly defines an "object" within the context of object-oriented programming. This, of course, assumes that Paone is describing an object that is defined in an object-oriented programming language. As the Court has found that no such limitation applies to this claim term, these definitions are not especially useful.

The Court notes that Microsoft objects that the Special Master improperly discounted these dictionary definitions because he wrongly believed that the field of the '789 patent is only cryptography, to the exclusion of computer science. Whether or not this is an accurate characterization of the Special Master's Report—at least one quotation from the Special Master's Report that Microsoft offers in support of this thesis does not appear to actually exist, (see Microsoft's Objections at 5)—the point is immaterial. Microsoft's definitions explicitly require that the field of the invention is object-oriented programming, not computer science. Such is not the case.

Moreover, while these technical dictionaries may provide a rough and ready definition of "object" within object-oriented programming, the Court has already noted that even experts do not appear to agree on a precise definition for either object-oriented programming or its components. See Mansfield at 264; see also TiVo, 516 F.3d at 1306 ("Tivo's expert offered a broader definition for the term 'object' . . [and stated] that objects and object-oriented techniques can be used in

many programming languages."). Likewise, contrary to the characterization of objects as "self-contained", one of the chief benefits of object-oriented programming—aside from data protection—is the use of "classes" and "inheritance", which are ways for multiple objects to reuse the same programming code. See Shammas at 11, 14 ("Object-oriented programming supports code reuse in a manner not available in structured programming."); Weisfeld at 14 ("Be aware that there is not necessarily a physical copy of each method for each object. Rather, each object points to the same physical code."). Thus, while it may be a beneficial conceptual definition in some spheres to define objects in object-oriented programming as "self-contained", this does not appear to be a strict technical requirement even within an object-oriented programming language.

Therefore, the Court accepts in part and rejects in part the Special Master's recommendation with respect to the construction of "object key", and construes the term as follows:

An "object key" is an encryption key that is not available to the general public, and which is composed of both (1) key data and (2) methods that modify that key data. The object key need not be implemented in an object-oriented programming language.

D. Construction of the Claim Term "Random Session Object Key"

The definition of the second term that the Court construes, "random session object key", follows in part from the definition of "object key." In the '789 Patent, the random session object key is an additional key that provides supplementary data

to the object key to aid in the object key's modification of its state. The term appears in multiple claims and throughout the specification. Typical of the term's use are claims 1 and 23, which refer to "modifying the at least one object key based on seeding from the random session object key" and to an "encryption/decryption process [that] further includes the use of a random session object key having an initial state randomly generated by the computer system, and wherein the object key modifications are based on seeding from the random session object key."

While the parties generally continue to dispute the import of the words "object key" in this term, the primary new dispute between the parties with respect to this term concerns the meaning of **the** word "session". The parties and the Special Master defined "random session object key" as follows:

Paone	**Microsoft**
"data with a random initial state and methods that operate on the data" [Does not object to the Special Master's recommendation.]	"'object key' (defined above) that is randomly generated for a communications session" [Objects to the Special Master's recommendation.]
Special Master Peterson	
The term "random session object key" was coined by the inventor and refers to a second encryption key, which is distinct from the "object key" or first encryption key. Both the "object key" and "random session object key" are "dynamic," i.e., both are modified from an initial static state. But that modification is required by other claim language. Also, the "random session object key" or second encryption key is used to modify the "object key," but that too is required by other claim language.	

In construing "random session object key", the Special Master found that the term was, like "object key", a purely coined term, and that the words "random", "session", and "object" in the term therefore need not be given any special meaning.

After reviewing the relevant intrinsic and extrinsic evidence, the Special Master concluded that the "random session object key" was essentially a "second encryption key," as distinguished from the "object key", which he called the "first encryption key". He noted that, like the "object key", the "random session object key" was dynamic, but that this quality was described by other claim language.

For his part, while Paone apparently gave meaning to at least the word "random" in his initial construction of "Random Session Object Key", he does not object to the Special Master's construction of the term. Microsoft, on the other hand, objects to the entirety of the Special Master's construction.

In objecting to the Special Master's construction, Microsoft first asserts that the grammar of "random session object key" requires that the term be construed to be a type of "object key", and contends that the Special Master's construction does not achieve this. Microsoft also maintains that the word "random" must be given meaning in the term, and thus requests that "random session object key" be defined to have randomly generated data. Finally, and most centrally, Microsoft urges that the word "session" must be interpreted to define "random session object key" as being randomly generated for a "*communications* session."

In support of this last assertion, Microsoft relies primarily on references to the word "session" in prior art and in a textbook, each of which explain that a "session" is a "communications session." In response, Paone notes that a number of the claims that reference the random session object key make no mention of communications usages, and at least one described embodiment of the invention involves encrypting

data for storage on a single computer—an application that necessarily does not involve communication. Paone also presents a number **of** definitions of "session" from dictionaries that define the word more broadly, as: "the time during which a program is running", Computer Dictionary: Third Edition, p. 337 (Microsoft Press 1997), "a period devoted to an activity", The Oxford Dictionary and Thesaurus: American Edition, p. 1025 (Oxford University Press 1996), and "one or more instances of an application", S.M.H. Collin, Dictionary of Personal Computing and the Internet, p. 169 (Fitzroy Dearborn Publishers 1997).

First, with respect to the central question of whether the word "session" means "communications session", the Court has reviewed the Special Master's recommendation that the Court reject this limitation, and finds his analysis to be without error. As the Special Master points out, nothing in the patent itself indicates that "session" must mean "communications session." Moreover, while it is not an inviolable rule, it is generally appropriate to interpret claim language to cover all specified embodiments of the invention. See, e.g., On-Line Tech v. Bodenseewerk Perkin-Elmer, 386 F.3d 1133, 1138 (Fed. Cir. 2004) ("a claim interpretation that excludes a preferred embodiment from the scope of the claim is rarely, if ever, correct" (internal quotations and citations omitted)). Here, the '789 patent specification includes an embodiment that involves no communication. For these reasons and the further reasons set forth in the Special Master's Report, the Court finds that the term "session" does not imply "communications session."

Nevertheless, the Court respectfully disagrees with the Special Master's conclusion that the words "random", "session", and "object" import no particular meaning to the term "random session object key."

First, for the reasons explained above, the Court finds that "object key" has a special meaning that derives in part from the word "object" and in part from that word's use in context. Thus, the Court now also finds that a *random session* object key incorporates the meaning of the term "object key". The Court concludes this not only from basic rules of grammar, but also from the use of "random session object key" in the claim language. For example, claim 13 describes "a computer implemented as defined in claim 3, wherein the modification method of said random session object key includes a hashing function." ('789 Patent, 13:28–30.) Claim 3, the claim on which claim 13 is dependent, introduces the "random session object key" as being involved in the encryption method, but never defines it as having "modification methods." However, the language of claim 13 suggests that Paone assumed that the reader would infer from the definition of "object key" that a "*random session* object key" also comprised both (1) data and (2) methods to modify that data. Likewise, in describing the operation of the random session object key in the preferred embodiment of the invention, the specification states, "[a]fter each block has been processed, the key modification methods in the object key [] and the random session object key [] are performed to modify the object key's and the random session object key's state." ('789 Patent 7:4–8.) Again, this description

assumes a parallelism between the structure of the object key and the structure of the random session object key.

Assuming, then, that "random session object key" is a type of object key, the Court further accords meaning to the words "random" and "session" as describing this type of "object key". As for "random", both Paone and Microsoft agree that the random session object key has data that is initially generated randomly. With respect to "session", the Court has already rejected Microsoft's construction that "session" means "communications session". In place of this definition, the Court applies the more generic definition of this term suggested by Paone, "one or more instances of an application", which the Court finds encompasses all of the described embodiments of the invention, and is consistent with the claim language itself. Therefore, the Court construes "random session object key" as follows:

A "random session object key" is an object key, as defined above, with data that is generated randomly for each instance of the use of the encryption application.

E. Construction of the Claim Term "Repeating the Step[] of Modifying the at Least One Object Key"

The Court next construes the claim term "repeating the step[] of modifying the at least one object key." This term appears in claims 1 and 32, in a form that for the present purposes is substantially identical. The relevant language in claim 1 reads:

> **1.** A computer implemented method for encrypting data comprising the steps of:
>
> > creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data;

> [creating a key schedule using the object key, encrypting a random session object key, and encrypting a block of plaintext using the key schedule]
>
> modifying the at least one object key based on seeding from the random session object key;
>
> modifying the key schedule based upon the at least one modified object key;
>
> encrypting a next block of input plaintext data utilizing said modified key schedule; and
>
> repeating the steps of modifying the at least one object key, modifying the key schedule and encrypting utilizing the modified key schedule until the encrypting of blocks of plaintext data is completed.

('789 Patent, 11:17–39 (emphasis added).)

The parties' primary dispute over this term is whether the disputed claim language implies that, for each data block, the object key must be modified from its *present state*, or whether it may be modified each time from its *original state*. The parties and the Special Master construed this term as follows:

Paone	**Microsoft**
Requires no construction. [Does not object to the Special Master's recommendation.]	"further modifying" the "at least one object key" [Objects to the Special Master's recommendation.]
Special Master Peterson	
The claim language does not exclude modifying the original "object key".	

In construing this claim term, Paone asserts that the disputed claim language is clear on its face, and that it plainly encompasses modification of the *original* object key. By contrast, Microsoft asserts that the claim language is ambiguous, and that this ambiguity must be resolved by looking to the preferred embodiment of the

invention—which describes the object key as being modified in each case from its *present* state. Microsoft further contends that this is the only reading of the claim that is consistent with the general described purposes of the invention and the prosecution history.

Ultimately, the Special Master ruled in favor of Paone on this issue, finding that the language of the claim did not exclude the repeated modification of the original object key, and that the specification did not show otherwise. For the reasons that follow, the Court respectfully disagrees with the Special Master's construction of this term, and finds that the disputed language limits the invention to modifying of the object key's present state.

The Court begins with the claim language itself. As relevant to the presently disputed term, claim 1 describes (1) the creation of an "object key", (2) the modification of the object key, and (3) the "repeating [of] the step[] of modifying the at least one object key." For the purposes of the following discussion, the Court will refer to these steps respectively as Step 1, Step 2, and Step 3. The Court construes all three steps to the extent necessary to resolve the parties' dispute over the language describing Step 3.

Beginning with the language of Step 1, "creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data", the Court construes this language to indicate that the "object key" that is modified in Step 2 is at least initially a newly created object key.

Step 2 then describes "modifying the at least one object key based on seeding from the random session object key". With respect to this language, the Court notes that it has previously construed the methods that are comprised in the object key to be methods that modify the object key. For the same reasons that underlie that construction, the Court now conversely construes Step 2 to describe modification of the object key by operation of the object key's own modification methods.

The disputed claim language of "repeating the step[] of modifying the at least one object key" then describes Step 3. Read most literally, this claim language supports Paone's construction. That is, Step 2 describes modification of the object key that is created in Step 1. Thus, when Step 3 essentially directs that Step 2 be repeated as written, it at least arguably describes a repeated modification of that same object key created in Step 1, based on that object key's state *as it existed in Step 1*.

Paone further asserts that any construction of Step 3 that limited modification of the object key to being based on the object key's *already modified* state would result in rewriting claims 1 and 32 as follows (the language that Paone asserts is being improperly read into these claims is underlined):

> **claim 1:** "modifying the at least one object key based on seeding from the random session object key and the object key after it has already been once modified" ('789 Patent, 11:17-38.)
>
> **claim 32:** "modifying the at least one object key using at least a nonlinear function and the object key after it has already been once modified" (Id. at 18:10-31.)

(Paone's Objections at 22.) Paone then asserts that:

> As illustrated in the claims, however, claim 1 requires basing the modification on "seeding from the random session object key," and

> claim 32 requires basing the modification on "at least a nonlinear function." Whether or not the modification step could also be based upon "the object key after it has already been once modified," the initial state of the object key, or something else is simply not a limitation of the claims.

(Id. at 23.)

However, Paone's assertions highlight the first difficulty with his own construction of Step 3. In the Court's view, Paone's interpretation ignores the claim language in Steps 2 and 3 indicating that the object key is "modified". In the opinion of the Court, this language implies that the object key's data—whether in original or already modified form—is an input into the object key's new state. (See, e.g., "Modify", Merriam Webster Online Dictionary, available online at <http://www.merriam-webster.com/dictionary/modify> ("3a: to make minor changes in; 3b: to make basic or fundamental changes in often to give a new orientation to or to serve a new end <the wing of a bird is an arm modified for flying>") (accessed February 2, 2011)). Thus, at the very least, the claims must be read to require that the "modification step" be based on the object key's data itself. There is no limitation on *including* other inputs into the modification algorithm—and indeed, claim 1 and claim 32 require additional inputs—but the object key's own data is a required input to "modify" the object key.

This interpretation of the word "modify" is bolstered by the Court's previous construction of the "object key". As noted above, the Court construed an object key as functioning analogously to an object in object-oriented programming. Thus, the

object key's modification methods operate only on its own data—using whatever other inputs that are required or desired—to modify that data.

Given this, Paone's construction of Step 3 is only possible if the object key continues to exist in its original state *even after* it has been modified once. However, to have the object key exist both in its original state and a modified state is a contradiction in terms. Once the object key has been modified, it by definition no longer exists in its original state. Thus, Paone's construction of Step 3 requires reading an additional step into the disclosed method: either the object key must be copied before it is modified, or the modified object key must be returned to its original state before it is modified again. However, the claims set forth neither of these additional steps.

Thus, a more reasonable interpretation of the meaning of the disputed language for Step 3 is to describe modification of the already modified object key. While this reading may require a less literal understanding of the claim language, it avoids the greater error of reading entire unwritten steps into the claims.

The Court's conclusion is supported by the embodiments of the invention specified in the patent document. Thus, in the description of the preferred embodiment, Paone refers to the "running state", (Id., 2:60), of the object key, which, in the Court's view, suggests that the object key takes on successive forms, each based on a previous form. Later, the specification states that "[t]he object key [] contains key modification methods that mutate the key's state." (Id., 5:18–20.) This

is also consistent with the understanding that the object key repeatedly modifies its own, previously modified data.

The parties and the Special Master also focus on Figure 3 in the '789 Patent, which is a flow chart that illustrates the high level functioning of the preferred embodiment of the patent. Microsoft points specifically to the description of step 52 in this drawing, which provides that "the state of the object key [] is modified based on seeding from the random session object key." (Id., 5:52–55.) According to Microsoft, the use of the word "state" in this description is further evidence that the object key is modified based on its own previously-modified key data.

The Special Master is equivocal on the point, but in the Court's view, language in the specification that follows shortly after this quoted language buttresses Microsoft's argument. Describing a series of steps in Figure 3, the specification provides, "new states for the random session object key[], the object key[], and a new key schedule [] are created for each plaintext data block . . . ." (Id., 5:64–6:1.) The patentee thus describes the object key and the random session object key as taking on "new states," while constructing the sentence to expressly provide that the key schedule does not take on a new state, but is simply "new". It is difficult to determine why Paone would make this distinction unless he were at least indicating that the object key and random session object key, as opposed to the key schedule, are in each new instance based on previous versions of themselves.

This distinction also is at odds with Paone's suggestion that a different passage of the specification supports his construction of the disputed claim term.

Specifically, Paone cites to a sentence in the specification that states, "[s]imilar to the object key[], the block's key schedule is regenerated anew for each plaintext input block." ('789 Patent, 3:2–4.) According to Paone, this sentence discloses an embodiment in which the object key is—in contrast, it would appear, to the language discussed directly above—"new" for each data block. However, the paragraph that directly precedes the quoted language describes the object key's "initial state" and its "running state", which are terms that, in the Court's view, identify the object key as being successively modified. The patentee then states only that the key schedule is "similar" to the object key, in that the key schedule is generated anew for each data block. In the Court's view, creating a "new" key schedule for each data block is in fact similar to—although not the same as—successively modifying the object key for each data block. Thus, these portions of the specification are at least neutral, and possibly support the Court's construction.

To be sure, the specification is not conclusive, and in general, claims should not be limited to a patent's disclosed preferred embodiments. See, e.g., Agfa Corp. v. Creo Products Inc., 451 F.3d 1366, 1376 (Fed. Cir. 2006). Nevertheless, the specification does provide evidence that may be used to interpret the presently disputed term. See, e.g., Phillips, 415 F.3d at 1315 ("[the specification] is the single best guide to the meaning of a disputed term"). Thus, reading the patent as an integrated instrument, the Court finds that the most reasonable interpretation of the ambiguous language, "repeating the step[] of modifying the at least one object key," in claims 1 and 32 is to limit the disclosed modification of the object key to

modification of its own, previously modified, data. The Court therefore construes the claim term as follows:

The claim term "repeating the step[] of modifying the at least one object key" is understood to be limited in the sense that the object key's data, as it presently exists in the object key at each instance of modification, must be an input into the modification methods of that object key.

F. Construction of the Claim Term "Block"

The parties next dispute the meaning of the claim term "block", as it is used to describe groups of data to be encrypted or decrypted. As relevant to the allegedly infringed claims, the term appears in claims 1, 23, and 32. Its use in claim 1 is typical. Claim 1 describes the methods of the invention as including the steps of:

> encrypting a block of input plaintext data utilizing said key schedule;
>
> . . .
>
> encrypting a next block of input plaintext data utilizing said modified key schedule; and
>
> repeating the step[] of . . . encrypting utilizing the modified key schedule until the encrypting of blocks of plaintext data is completed.

('789 Patent, 11:26–38 (emphasis added).)

The primary dispute between the parties over this term is whether it requires that all blocks of data in a given encryption be of the same length, or whether block length may vary among blocks. The parties' and the Special Master's constructions are as follows:

Paone	Microsoft
"a fixed-length grouping of bits" [asserts that block length may vary from block to block; objects only to the portion of the Special Master's recommendation that rejects this assertion]	"string of fixed length [that applies to all strings] defined by a block cipher" [asserts that block length may not vary from block to block; does not object to the Special Master's recommendation]
Special Master Peterson	
The term "block" as used in the claims and specification of the '789 patent means "a sequence of bits wherein that sequence has a fixed length that does not vary from block-to-block." The length of a block may be determined through selection in an encryption algorithm. Plaintext data having a length longer than the length of a block is divided into blocks of fixed length.	

After reviewing the relevant intrinsic and extrinsic evidence, the Special Master ruled in Microsoft's favor, and held that the length of each block of data may not vary from block to block. The Court has reviewed this evidence *de novo*, and finds that the Special Master's analysis is without error. In addition, the Court has reviewed Paone's objections to the Special Master's Report regarding the definition of "block", and finds his objections to be without merit, for substantially the reasons set forth in the Special Master's Report.

In addition, the Court notes that an additional piece of intrinsic evidence apparently not addressed by the parties or the Special Master supports the Special Master's conclusion. The patent provides in claim 15 for "[a] computer implemented method as defined in claim 2, wherein input plaintext is compressed . . . and padded . . . to produce a file with a length that is evenly divisible by the block length . . . ." ('789 Patent, 13:36–41.) Claim 15's description of a file that is "evenly divisible by the block length" is a strong indication that, at least for that claim, block length does not vary from block to block. Otherwise, the term "evenly divisible"

would have little meaning. The Court is cognizant of the bar on reading the additional limitations of claim 15 into claims 1 and 2. However, claim 15 does not define "block" separately from claims 1 and 2, but rather inherits this term from them as a dependent claim. Moreover, claim 15 says nothing about the "block" or "block length" until it is referenced in a way that implies that a block has a fixed length. While this evidence is not conclusive, the Court finds that it does support the Special Master's finding that the term "block" as used in claim 1 has a fixed length. See Phillips, 415 F.3d at 1314 ("Because claim terms are normally used consistently throughout the patent, the usage of a term in one claim can often illuminate the meaning of the same term in other claims.")

Thus, for the reasons set forth in the Special Master's Report and the reasons set forth herein, the Court adopts in its entirety the Special Master's analysis of the construction of the term "block", and likewise adopts in its entirety the Special Master's construction of that term, which is:

> The term "block" as used in the claims and specification of the '789 patent means "a sequence of bits wherein that sequence has a fixed length that does not vary from block-to-block." The length of a block may be determined through selection in an encryption algorithm. Plaintext data having a length longer than the length of a block is divided into blocks of fixed length.

G. Construction of the Claim Term "Key Schedule"

The Court next construes the claim term "key schedule". Like "object key" and "random session object key", this term appears throughout the patent and represents one of the central elements of the invention disclosed in the allegedly infringed claims. The term's use in claim 1 is typical:

> **1.** A computer implemented method for encrypting data comprising the steps of:
>
> creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data;
>
> creating a key schedule based upon the at least one object key;
>
> . . .
>
> encrypting a block of input plaintext data utilizing said key schedule;
>
> modifying the at least one object key based on seeding from the random session object key;
>
> modifying the key schedule based upon the at least one modified object key;
>
> encrypting a next block of input plaintext data utilizing said modified key schedule; and
>
> repeating the steps of modifying the at least one object key, modifying the key schedule and encrypting utilizing the modified key schedule until the encrypting of blocks of plaintext data is completed.

('789 Patent, 11:17–38 (emphasis added).)

The parties' dispute over this term centers on whether the key schedule is required to be a list of multiple keys, or whether it may be a single, expanded key. The parties' and the Special Master's constructions of this term are as follows:

[Remainder of page left intentionally blank.]

Paone	Microsoft
"the result of an algorithm that expands a relatively short master key to a relatively large expanded key for use in an encryption or decryption algorithm" [Objects only to the portion of the Special Master's recommendation that describes an example of encrypting using the key schedule.]	"ordered list of keys" [Does not object to the Special Master's recommendation "provided it is taken in its entirety, and to the extent that it is appropriately adapted to the remainder of the asserted claims as suggested by the Master". (Microsoft's Objections at 30.)]
Special Master Peterson	
A "key schedule," as used in the claims and as described in the specification, is a term used by the inventor that refers to an algorithm that is used in encrypting a block of input plaintext data. The "key schedule" is created by using an object key as an input to an expansion function that increases the size. The result is called a "key schedule." In the step of "encrypting a block of input plaintext data utilizing said key schedule," i.e., the "key schedule" algorithm, and using claim 1 as an example, a plaintext block is encrypted with a first 64 byte portion of the key schedule, which might be referred to as KS 0, that 64 byte portion is moved to a different location along the bits of the key schedule, which might be referred to as KS 1, to create a different 64 byte portion of the key schedule which is then used to again encrypt the same plaintext block, and the process is repeated until the end of the key schedule is reached. The references to KS 0 and KS 1 are not limiting and are used solely for ease of reference. Also, the claims are not limited to 64 byte portions – those are solely for purposes of explanation.	

Before embarking on an analysis of the parties' dispute over this term, the Court first briefly discusses the functioning of the key schedule as described in the preferred embodiment of the invention.

Ignoring for the present purposes the random session object key, the preferred embodiment of the invention provides for a two-tier encryption key system, which comprises the object key and the key schedule. Under this system, the object key, which contains a relatively small number of 1's and 0's (again, called "bits"), is expanded using an algorithm to create the key schedule, which has a far greater number of bits. The key schedule, and not the object key, is then actually used to

encrypt the plaintext. Because the key schedule depends directly on the object key for its data and is regenerated based on the object key for each plaintext block, it is colloquially appropriate to say that the object key is used to encrypt each plaintext block. As a technical matter, however, the key schedule actually provides the data that is used directly in the encryption process.

In the preferred embodiment, the object key is 4,096 bits long. These 4,096 bits are then expanded to 13,584 bytes (or, since there are 8 bits in a byte, 108,672 bits) to form the key schedule. The preferred embodiment describes encryption of blocks of data that are 512 bits long, thus rendering the preferred-embodiment key schedule more than 200 times the length of a data block.

The specification then describes using this key schedule in a complex method to encrypt the 512-bit blocks of plaintext. To carry out this encryption, the invention employs various portions of the 108,672-bit key schedule to repeatedly perform multiple functions on the data block. Three of these functions merit brief description.

The first of the three functions is called an "S-Box" substitution. ('789 Patent, 6:57–59.) Explained in only the most basic terms, the disclosed S-Box substitution calls for 8-bit portions of the data block to be fed into a table that looks up another pre-arranged 8-bit string to replace the original data. Then, at various times in the encryption algorithm, parts of the key schedule are used to re-arrange the table, so that the same 8-bit input fed into the S-Box after it has been rearranged would produce a different output. In the preferred embodiment of the patent, substitution using the S-Box, as well as rearrangement of the table, happens multiple times.

The second function is referred to as a transposition. (Id., 6:66). Transposition involves rearranging the 512 bits of the data block into a different order. For this function, various portions of the key schedule are used to dictate where each bit is to be re-located within the data block. In the preferred embodiment, this function is similarly carried out numerous times during the encryption of a single data block, using different portions of the key schedule at different times.

The third function is called a bit-wise "exclusive or", and is often abbreviated "XOR". (Id., 6:64.) In a bit-wise XOR function, one bit from the data block is compared to one bit from the key schedule. If both bits are *the same* (both 1's or both 0's), then the bit from the data block is replaced with a 0. If the bits are *different* (one 1 and one 0, in either order), then the bit from the data block is replaced with a 1. The output of an XOR function is thus as follows:

Data block input	**Key stream input**	**Data block output**
0	0	0
0	1	1
1	0	1
1	1	0

When an XOR function is carried out, each bit from the data block is compared with a different bit from the key schedule. In the preferred embodiment, this is done a total of sixteen times for the entire data block, using different sections of the key schedule each time that the XOR function is carried out. (See '789 Patent, Fig. 5A.)

Having reviewed the preferred embodiment of the invention in greater detail, the Court now returns to the construction of the term "key schedule". Both Paone and Microsoft essentially agree that the key schedule, as the term is used in the claims, is produced by expanding the object key. Prior to the issuance of the Special Master's Report, Paone had insisted on referring to the "object key" in this sense as a "master key", but he appears to have abandoned this position in light of the Special Master's recommendation. Likewise, prior to the Markman hearing, Microsoft construed the key schedule as a "table" of keys, but has since modified that construction to be an "ordered list" of keys. Thus, at the conclusion of the Markman hearing, the primary dispute between the parties concerning the construction of the key schedule was whether it should be defined as a single, long, key, or whether it should be defined as a list of multiple keys, maintained in an "ordered list."

In his Report, the Special Master rejected both of the parties' proposed definitions, and offered instead the following recommendation, which the Court has divided into two sections for ease of discussion. The first part of the Special Master's Recommendation describes the formation of the key schedule, and reads:

> A "key schedule," as used in the claims and as described in the specification, is a term used by the inventor that refers to an algorithm that is used in encrypting a block of input plaintext data. The "key schedule" is created by using an object key as an input to an expansion function that increases the size. The result is called a "key schedule."

(Special Master's Report at 181). The second part of the Special Master's recommendation then provides a description of encryption using the key schedule, and reads:

> In the step of "encrypting a block of input plaintext data utilizing said key schedule," i.e., the "key schedule" algorithm, and using claim 1 as an example, a plaintext block is encrypted with a first 64 byte portion of the key schedule, which might be referred to as KS 0, that 64 byte portion is moved to a different location along the bits of the key schedule, which might be referred to as KS 1, to create a different 64 byte portion of the key schedule which is then used to again encrypt the same plaintext block, and the process is repeated until the end of the key schedule is reached. The references to KS 0 and KS 1 are not limiting and are used solely for ease of reference. Also, the claims are not limited to 64 byte portions – those are solely for purposes of explanation.

(Id.)

Both Microsoft and Paone at least conditionally accept the first part of the Special Master's recommendation. However, Paone directly rejects the second part of the Special Master's recommendation, and Microsoft states that it accepts the first section only insofar as the second section is also adopted and applied by the Court.

The parties' divergent response to the second section of the Special Master's recommendation reveals the substantive basis for the parties' dispute over "key schedule". At the Markman hearing, the parties discussed in some detail the disclosed preferred embodiment of the key schedule and its use as illustrated in Figure 5A of the '789 Patent. Both parties agreed that, in the preferred embodiment, various sections of the key schedule are used at different times to carry out various encryption functions. Microsoft asserted that the key schedule is therefore made up of various keys that are listed in order, so that the encryption algorithm can access those keys when necessary. Paone countered that the key schedule was simply one long string of bits, and that to the extent that the encryption algorithm called for using

only some of those bits at a time, the required bits could be accessed simply by identifying their location in the string.

In resolving this dispute, the Special Master largely followed the construction Paone presented at the Markman hearing. (See Markman Hearing Tr. at 81–82.) Thus, the Special Master held that that bits in the key schedule can be identified by their position in the key string, and that the key schedule need not be an ordered list. (See, e.g., Special Master's Report at 168.) This particular holding is described in the second part of the Special Master's recommendation on this term. However, despite the Special Master's adoption of Paone's proposed construction, it is *Microsoft* that accepted the second part of the Special Master's recommendation, not Paone.

Why should this be the case? Primarily, it is because this second part of the Special Master's recommendation also describes using the key schedule in an encryption process that is *iterative*. That is, while noting that parts of the key schedule could be identified by their position, the Special Master also stated that, to encrypt a single block of plaintext, the same encryption method is repeated multiple times on the same data block. This distinction between iterative (that is, repeated function) block ciphers and non-iterative (that is, single function) block ciphers is the crux of the parties' true dispute. As described in detail below for the construction of the term "block cipher", Microsoft asserts that the disclosed invention is an iterative block cipher, and Paone rejects this limitation. Microsoft thus accepts the second section of the Special Master's recommendation for "key schedule" because it describes an iterative block cipher. Paone rejects it for the same reason.

This is not to say that the construction of "key schedule" is unrelated to the parties' underlying dispute over the term "block cipher". When its argument is reduced to its essentials, Microsoft would appear to assert that the term "key *schedule*" implies that the data in the key schedule cannot be used all at once, but must be used in pieces. Otherwise, the word "schedule" would have no meaning; "schedule" implies sub-parts, and a key schedule that was used all at once would have been simply named the "expanded key". Microsoft also appears to conclude from this that using the key schedule in pieces implies an iterative method of encryption. For his part, Paone reads no such limitation into the claims. He asserts that the key schedule may be used in parts by identifying various sections of the key by their locations, as in the preferred embodiment of the invention. However, Paone implies that nothing would limit the use of the key schedule as a whole (or just in part) to fully encrypt a single block of data using just one encryption function.

Having reviewed the evidence and the parties' submissions, the Court agrees with the Special Master that the key schedule need not be construed as an "ordered list" of keys. The Court also agrees that portions of the key schedule must be accessible for independent use in a given encryption function, and that the various parts of the key schedule may be accessed simply by identifying their position in the string of data that makes up the key schedule.

As for the Special Master's example of encryption using the key schedule, the Court finds that, while this appears to be an acceptable embodiment of the invention disclosed in claim 1, this description does not limit this or any other claim. Most

critically, this is because this limitation does not encompass the preferred embodiment of the invention as disclosed in the specification. In his example, the Special Master describes a method of encryption whereby each bit of the key schedule is compared with one bit in the data block. This is essentially a description of encrypting using repeated applications of the bit-wise XOR function (or another similar bit-by-bit comparison encryption function). However, in the preferred embodiment of the invention, where the data block is 512 bits long and the key schedule is 108,672 bits long, this would require over 200 XOR iterations to compare each bit of the key schedule to a bit in the data block. The preferred embodiment discloses only 16 XOR functions for each data block.

More importantly, the preferred embodiment also discloses functions such as the S-Box substitution, which does not involve comparing key schedule data to plaintext data, but rather involves using key schedule data to re-arrange a pre-existing substitution table. Similarly, the preferred embodiment discloses a transposition function as well. This function also does not compare the key data to the plaintext data, but rather uses the key data to direct the bits of the plaintext block into a different order. In addition, for both of these functions, the number of key data bits used for a single implementation of the function almost certainly would not match the size of the plaintext block, as the Special Master's example requires. The Special Master's example is therefore rejected as not encompassing the embodiments disclosed in the specification. See, e.g., On-Line Tech v. Bodenseewerk Perkin-Elmer, 386 F.3d 1133, 1138 (Fed. Cir. 2004) ("a claim interpretation that excludes a

preferred embodiment from the scope of the claim is rarely, if ever, correct" (internal quotations and citations omitted)). As for the question raised by this example of whether the invention disclosed in the '789 patent must be an iterative block cipher, the Court addresses this below in its construction of "block cipher".

With respect to the issue of whether the term "key schedule" implies that the data in the key schedule must be used in pieces rather than as a whole, the Court notes that neither party expressly raised or briefed this issue. Moreover, the Court finds that this implicit dispute derives from the parties' underlying disagreement over the term "block cipher". Thus, the Court declines to directly resolve this issue, as it does not appear to be necessary to aid the jury.

Finally, the Special Master's recommended construction of "key schedule" refers to the key schedule as an "algorithm". The Special Master appears to have adopted this construction from certain references in the prior art that refer to "key schedules" as not merely the result of an expansion algorithm, but as themselves algorithms for encryption. However, while neither party objects to the Special Master's characterization of "key schedule" as an algorithm, neither party urged this construction prior to the time that the Special Master issued his Report. Moreover, in reviewing the claims and the specification, the Court finds no indication that the key schedule is an algorithm for encryption. Rather, the patent language suggests that the key schedule merely supplies data to be used by the encryption algorithms in other parts of the invention. The Court thus finds no basis for construing the key schedule as itself a method or algorithm for encryption.

Thus, the Court construes the term "key schedule" as follows:

The "key schedule" is a string of bits that is used in encrypting a block of input plaintext data. It is created by using an object key as an input to an expansion function, which increases the size of the object key to form the key schedule. Sub-portions of the key schedule must be accessible to be used separately in the encryption algorithm if called for, but those sub-portions may be accessed merely by identifying those sub-portions' positions in the string of bits that forms the key schedule.

H. Construction of the Claim Term "Block Cipher"

Finally, the Court construes the claim term "block cipher". This claim term appears in the patent name, as well as throughout the claims and the specification. With respect to the alleged infringed claims, it appears in claims 1, 23, 26, and 32. The use of the term in claim 1 is again typical:

> **1.** A computer implemented method for encrypting data comprising the steps of:
>
> > creating at least one object key in a block cipher, the at least one object key comprising data and methods that operate on said data;
> >
> > . . .
> >
> > encrypting a random session object key in a block cipher encryption process with the at least one object key . . . .

('789 Patent, 11:17–26 (emphasis added).) As previously intimated, the parties' primary dispute over this term is whether the term "block cipher" implies that the invention is an "iterative block cipher". However, as discussed below, there is also an underlying dispute as to the distinction between a "block cipher" and a "stream

cipher". The parties' and the Special Master's constructions of this term are as follows:

Paone	Microsoft
"a cipher that transforms a string of input bits of fixed length into a string of output bits of fixed length" [Does not object to the Special Master's recommendation.]	"iterative block cipher" which, in turn, means "scheme that breaks up a plaintext message into strings of fixed length and then repeatedly applies different keys to each string." [Objects to the entirety of the Special Master's recommendation.]
Special Master Peterson	
In view of the foregoing, the master recommends that the Court conclude that "block cipher" in the claims of the '789 patent is not limited to "iterative block cipher."	

Before construing this claim, the Court provides some additional general description of computer implemented symmetric key ciphers.

The Court has previously noted that on a very basic level—and perhaps an oversimplified level—a block cipher encrypts several bits of data at the same time, while a stream cipher encrypts only one bit of data at a time. The Court has also noted that an iterative block cipher encrypts a single block of data using the same encryption functions multiple times, while a non-iterative block cipher does not repeat encryption functions.

One of the limitations of a stream cipher is that it encrypts data by performing only one, simple encryption function on each element of data. (See, e.g., Menezes, Alfred, et al., Handbook of Applied Cryptography, p. 191 (CRC Press 1997) ("[Stream ciphers] encrypt individual characters (usually binary digits) of a plaintext message one at a time").) Most often, this single function is a bit-wise XOR function.

Given this limitation, stream ciphers must derive their security from having a non-predictable stream of key data to be compared to the plaintext data. Effective stream ciphers often do this by using sophisticated algorithms, which may even be based on previously encoded bits of the plaintext data. See, generally, Huth, Michael, Secure Communicating Systems, pp. 81–95 (Cambridge University Press 2001); Talbot, John, Complexity and Cryptography, pp. 99–115 (Cambridge University Press 2006); Robshaw, M.J.B., Stream Ciphers: RSA Laboratories Technical Report, TR-701 (RSA Laboratories 1995) (available online at <http://security.ece.orst.edu/koc/ece575/rsalabs/tr-701.pdf> (accessed February 3, 2011)).

By contrast, block ciphers have the capacity to perform a much more diverse set of encryption functions on plaintext data. As illustrated by the '789 patent's specification, a block cipher need not simply execute bit-by-bit comparisons of plaintext data to key data. A block cipher may instead, for example, transpose (that is, rearrange) the bits within the block based on the key data, or use a substitution function to replace sets of bits with new sets of bits. In addition, a block cipher may iterate any or all of these functions, so that, for example, not just one XOR is performed on the key data, but 20 rounds of XOR are performed. The ability to perform (1) varied and (2) repeated encryption functions permits a block cipher to be secure without necessarily having a complex method for deriving a long string of key data.

One of the benefits of the '789 patent subject matter—at least as it appears in its preferred embodiment—is that it combines the positive attributes of both of these

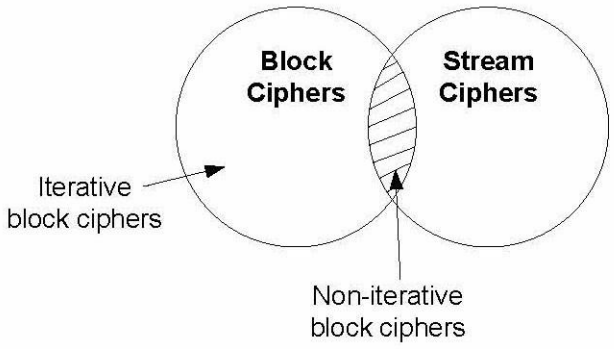
types of ciphers. Just like a stream cipher, the key data in Paone's invention is constantly changing in a way that is intended to be difficult to predict. However, the invention is also a block cipher, so that this ever-changing key data need not be compared just once to plaintext data to produce ciphertext, but may be used to encrypt data using other functions, and may also repeat those functions on a single block of data.

With this background, the Court returns to the construction of the term "block cipher". At least on its surface, Microsoft's argument in favor of construing "block cipher" as "iterative block cipher" relies primarily on Paone's statements to the various patent examiners during prosecution and re-examination. In a number of instances, Paone referred to his invention as an "iterative block cipher", and sometimes he did so in very direct ways, stating, for example, "[t]he present invention is directed to an iterative block cipher encryption method . . ." in his response to the PTO's office action of September 29, 1999. (Microsoft's Claim Construction Brief, Ex. C, MSLP0854424.)

In his Report, the Special Master reviewed the relevant prosecution history in detail, as well as binding Federal Circuit law, concerning the effect of Paone's statements to the PTO. Considering the evidence and the law, the Special Master concluded that Paone had not disclaimed non-iterative block ciphers in his submissions to the PTO. The Court has reviewed the relevant facts and law *de novo*, and agrees with the Special Master on this issue to the extent that he found that Paone

never explicitly disclaimed coverage of non-iterative block ciphers. Therefore, in this limited regard, the Court adopts the Special Master's analysis.

However, this does not fully resolve the construction of the disputed term. Microsoft makes two more arguments in favor of its construction that are only tangentially related to Paone's statements to the PTO. First, Microsoft asserts that by disclaiming all stream ciphers—a fact about which there is no dispute (see, e.g., Markman Tr. at 101–02 (Counsel for Paone stating: "we're not running away from the statement that this patent is not addressed [to] stream cyphers, it's not. Obviously it's a block cypher which is expressly set forth in the claims."))—Paone also by definition disclaimed all non-iterative block ciphers. To reach this conclusion, Microsoft asserts that *all* non-iterative block ciphers are, in fact, also *stream ciphers*. In its objections to the Special Master's report, Microsoft provided the following Venn diagram to illustrate its point:



(Microsoft's Objections at 23.) Microsoft then concluded that, by disclaiming *all* stream ciphers, Paone "disavowed coverage of all block ciphers that fall within the striped overlapping region"—that is, he disclaimed coverage of all non-iterative block ciphers. (Id. at 24.)

More than anything else, this appears to be the heart of Microsoft's argument in favor of its construction of this claim term. At the Markman hearing, Microsoft's attorney asserted that Paone claims that the '987 Patent covers an encryption method that employs only a single round of bit-by-bit comparison of plaintext data to key data for each block. Microsoft's attorney then stated:

> Now, what's wrong with this picture? In the patent office, Mr. Paone's attorney said, Our invention is an iterative block cypher and not a stream cypher. And yet the construction that they're advancing is such that it can read on a stream cypher because it's one that encrypts one bit at a time in an XOR operation. There is the rub in this case because they are pointing to a cipher in TKIP and BitLocker that they say is a block cipher that, in fact, encrypts exactly as a stream cipher encrypts. And that's what they told the patent office their invention was not. And they say our invention [] specifically defines an iterative block cypher encryption method and doesn't encompass a stream cypher. And there's the problem in the case.

(Markman Tr. at 97–98.)

However, Microsoft's argument mischaracterizes the state of the art in computer-based encryption, and is refuted with a simple counter-example. As noted before, the '789 Patent itself disclosed a number of encryption functions that could not be used in a stream cipher, but which also do not require iteration to be used. For example, a cipher that transposes—that is, trades positions among—the elements of a block of data *just one time* would be a non-iterative block cipher that is *not* a stream cipher. Such an algorithm (1) is a block cipher, because it encrypts a block of data; (2) is non-iterative, because it uses only one round of encryption; and (3) could not be carried out by a stream cipher, because it requires encryption of a number of elements of the data block at the same time. (See, e.g., Menezes at 191 ("[Stream ciphers]

encrypt individual characters (usually binary digits) of a plaintext message one at a time"); see also Schneier at 347 ("Confusion is enough for security [in a block cipher]. An algorithm consisting of a single key-dependent lookup table of 64 bits of plaintext to 64 bits of ciphertext would be plenty strong[, but this type of single round encryption is impractical because it requires a great deal of computer memory]").) Thus, non-iterative block ciphers are not necessarily also stream ciphers.

Perhaps aware of this weakness, Microsoft makes its final argument, and proposes an alternative definition of "block cipher". This alternative definition, which Microsoft asserts will nevertheless still exclude all stream ciphers, is:

> "block cipher" means "a cipher that encrypts plaintext data block-by-block but not one bit at a time"

(Microsoft's Objections at 26.)

This definition is compelling. As discussed above, the essence of the difference between a block cipher and a stream cipher is that a block cipher may both (1) use additional encryption functions that are unavailable to a stream cipher, and (2) repeat functions multiple times within a block of plaintext. A block cipher that takes advantage of *neither* of these benefits, and encrypts blocks of data using only a single bit-wise comparison between the plaintext and the key, is only a block cipher in the loosest sense. Although such a cipher might not output the ciphertext until all elements of the data block have been encrypted, each element of the data block is fully encrypted upon its comparison to an element of key data. As such, this type of cipher might be better referred to as a "delayed output stream cipher". The only practical difference between this type of algorithm and a pure stream cipher is that the

fully encrypted bits in this so-called block cipher are not produced as an output until the rest of the bits in the block have been encrypted. However, the Court can discern no effect that this distinction has on the substantive encryption method or the security of the cipher. Thus, to the extent that such a cipher may be defined as a "block cipher", it is also a "stream cipher".

In a response to the PTO dated December 29, 1999, Paone distinguished his patent subject matter from a stream cipher. He stated that his invention "defines an iterative block cipher encryption method," and that "[stream] ciphers and iterative block ciphers are substantially different in their function, use and design." (Microsoft's Claim Construction Brief, Ex. C, MSLP0854429.) In the Court's view, a cipher that encrypts each block of data using only a single bit-by-bit comparison between plaintext and key data is at least within the purview of what Paone disclaimed in this communication.

Therefore, the Court defines the term "block cipher" as follows:

A "block cipher" is, with the exception noted below, a cipher that encrypts data in blocks. A block cipher need not be iterative. However, a cipher that encrypts data in blocks, but does so by performing a single, successive, comparison of each element of the data block to an element of key data, is not a block cipher as the term is used in the '789 Patent.

III. CONCLUSION

For the foregoing reasons, the Special Master's Report and Recommendation is accepted in part, modified in part, and rejected in part as set forth herein, and the

disputed claim terms are construed to have the meanings set forth above. The Court graciously thanks Special Master Peterson for his outstanding service, and respectfully directs the Clerk of the Court to terminate the Special Master's appearance in this case.

SO ORDERED.

Dated: Central Islip, New York
February 9, 2011

/s/ Arthur D. Spatt
ARTHUR D. SPATT
United States District Judge